Activation-Based Weight Significance Criterion for Pruning Deep Neural Networks

Jiayu Dong[1,2,3], Huicheng Zheng[1,2,3(✉)], and Lina Lian[1,2,3]

[1] School of Data and Computer Science, Sun Yat-sen University, 135 West Xingang Road, Guangzhou 510275, China
zhenghch@mail.sysu.edu.cn
[2] Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, Sun Yat-sen University, 135 West Xingang Road, Guangzhou 510275, China
[3] Guangdong Key Laboratory of Information Security Technology, Sun Yat-sen University, 135 West Xingang Road, Guangzhou 510275, China

Abstract. Due to the massive amount of network parameters and great demand for computational resources, large-scale neural networks, especially deep convolutional neural networks (CNNs), can be inconvenient to implement for many real world applications. Therefore, sparsifying deep and densely connected neural networks is becoming a more and more important topic in the computer vision field for addressing these limitations. This paper starts from a very deep CNN trained for face recognition, then explores sparsifying neuron connections for network compression. We propose an activation-based weight significance criterion which estimates the contribution that each weight makes in the activations of the neurons in the next layer, then removes those weights that make least contribution first. A concise but effective procedure is devised for pruning parameters of densely connected neural networks. In this procedure, one neuron is sparsified at a time, and a requested amount of parameters related to this neuron is removed. Applying the proposed method, we greatly compressed the size of a large-scale neural network without causing any loss in recognition accuracy. Furthermore, our experiments show that this procedure can work with different weight significance criterions for different expectations.

Keywords: Pruning · Network compression · Weight significance

1 Introduction

Since the breakthrough in [1], deep convolutional neural networks have become the state-of-the-art techniques for various computer vision tasks, especially for image classification and recognition problems. Different architectures [1–5] have been proposed over the years and achieved better and better classification or recognition accuracy. One of the main trends of improving the network architectures is by increasing their depth, adding more layers to the network structures.

Y. Zhao et al. (Eds.): ICIG 2017, Part II, LNCS 10667, pp. 62–73, 2017.
https://doi.org/10.1007/978-3-319-71589-6_6

While improving the classification or recognition results, a deeper network always requires higher computational complexity and larger disk memory, which in a way hinders its utilization in certain scenarios, such as embedded systems or real-time applications.

Face recognition is one of the most challenging problems in image processing and computer vision. Many excellent works [13,17,19] have successfully applied deep CNNs to tackle face recognition tasks. Massive parameters ensure the ability of deep CNNs to express and discriminate complex face images. And deeper architectures are often useful in learning better and more abstract features. Another critical factor to the success of deep CNNs in face recognition is their capability to exploit large training datasets, which sometimes may be difficult to obtain, though. Meanwhile, these factors also contribute to their rapidly growing demand for model storage and computational resources.

To balance recognition performance against the size of network models, we can first train a densely connected baseline model to learn good face features, and then prune redundant connections to perform compression while preserving good recognition accuracy as much as possible.

Sparsifying connections is often performed in fully-connected layers, in which weights are sorted according to some significance criterions, and those weights with less significance are pruned. Such criterions are sometimes defined in terms of saliency. In this paper, we propose an activation-based criterion to sort the connections by their contributions in the activations of the next layer. Our pruning procedure sparsifies one neuron at a time, pruning a certain amount of connections of a neuron and then continues pruning the next one until the expected compression rate is reached. It also includes a surgeon step, in which whenever weights are removed in a pruning step, the remaining weights would be adjusted to maintain the whole network status. Therefore, after a model is pruned, the retraining or fine-tuning process is not requested. However, many observations including ours show that fine-tuning the pruned models can further improve their recognition performance, which would be verified by experimental results in this paper.

The main contributions that we made in this paper are as follows: (1) we propose a simple and yet effective weight pruning procedure, which performs well in compressing deep convolutional neural networks with large parameter sets; (2) we propose an activation-based weight significance criterion, which indicates that weights with less contribution to the activations of neurons in the next layer should be removed first; (3) The pruning procedure that we devise can be combined with different weight significance criterions for different performance improvements, such as higher compression rate or better recognition results.

The rest of this paper is organized as follows. Section 2 introduces some related works in sparsifying deep neural networks. The proposed method is described in detail in Sect. 3. The experimental results are displayed and analyzed in Sect. 4. Finally, we conclude this paper by Sect. 5.

2 Related Work

The motivation of pruning densely connected networks is that it helps to reduce the model size and therefore make it easier to implement deep neural networks in certain scenarios like embedded platforms with limited disk memory and computational resources. And also in many studies like [7,9], it is believed that smaller and sparser models can address the over-fitting problem and lead to better generalization.

One of the most classical works of dropping weights in deep neural networks is done by LeCun *et al.* [6] and known as Optimal Brain Damage (OBD). The authors used second derivative information to evaluate the saliencies of weights and perform pruning. Optimal Brain Surgeon (OBS) [11] further calculates the second derivative by dropping some approximations made by OBD.

Instead of trying to minimize the growth of the objective function during pruning parameters like the above two methods, Srinivas and Babu [12] suggested removing those neurons in a layer whose absence cause the minimum change of the activations of the neurons in the next layer. They proposed a data-free pruning procedure, which prunes an entire neuron at a time whenever two neurons have similar weight sets. The saliency is then defined as:

$$S_{i,j} = \langle a_j^2 \rangle \|\varepsilon_{i,j}\|_2^2 \tag{1}$$

$S_{i,j}$ represents the saliency that is evaluated between two neurons i and j. And a_j is the parameter that connects neuron j to a neuron in the next layer, and $\langle a_j^2 \rangle$ denotes the average of the square of the scalar a_j over all neurons in the next layer, and $\|\varepsilon_{i,j}\|_2$ is the Euclidean distance between two weight sets corresponding to the two neurons.

In [13], the authors proposed a correlation-based criterion for weight pruning, which removes those weights that have small neural correlations first. Their method performs pruning in one layer at a time, and after a layer is sparsified, the pruned model needs to be retrained. Due to the lack of the surgeon operation, OBD also requires retraining after each pruning operation. In the cases of OBS and [12], they would adjust the rest of the parameters after each pruning step to maintain the objective function or activations in the next layer, so they do not need retraining or fine-tuning.

All of the above methods only consider the significance of weights when pruning and do not care about the structure of pruned networks. To avoid irregular network structures after pruning, [8] introduces a structure pruning method which fits the pruned network on parallel computation.

As another line of reducing the size of neural network models, [14–16] applied singular value decomposition, low rank matrices approximation or vector quantization of weight matrices of the network to perform compression. Only small amount of parameters are needed to represent or predict all the parameters of the deep architecture. These methods do not prune weights in the same way as the above methods. Rather than pruning them, they use the approximations of weight matrices by which they can store much fewer weights than the original networks.

In [10], they used pruning, trained quantization and Huffman coding together to deeply compress the neural network and achieved good results, which shows that pruning methods and numerical computation methods can be well implemented together to further reduce the model size. In this paper we focus on finding a good pruning strategy, including the weight significance criterion and pruning procedure. And our proposed method is demonstrated to be superior in the experiments.

3 Pruning Connections of Deep Neural Networks

We choose the classical VGG-16 architecture proposed in [17] as our baseline model, which starts from a series of convolutional layers and ends with three fully-connected layers. The fc6 layer and fc7 layer are two fully-connected layers following the last convolutional layer. The fc8 layer is a n-ways classifier whose dimension depends on training datasets.

The fully-connected layer fc6 and fc7 contain most of the parameters of the whole architecture, and many researches [12,13] also point out that most of the redundancy exists in fully-connected layers. Therefore, our pruning operation is performed in the fc6 and fc7 layers.

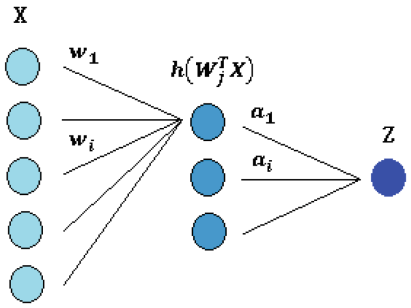


Fig. 1. Illustration of the forward propagation in a neural network.

3.1 Maintaining the Activations of the Next Layer

In [12], the authors proposed to remove neurons that cause the least change in the activations of the next layer. The weight matrix W connects the neurons in the previous layer to the neurons in the current layer. A neuron in the current layer is connected to the previous layer through a weight set W_i (i indicates the index of a neuron), and is connected to the next layer by a parameter a_i. The forward propagation process is demonstrated in Fig. 1. In order to make it easy to illustrate the process, let us assume there is only one neuron in the next layer, when there are two or more neurons, we average the values of a_i over all of the neurons. And the activation of the next layer can be written as:

$$z = a_1 h(W_1^T X) + \cdots + a_i h(W_i^T X) + \cdots + a_j h(W_j^T X) + \cdots + a_n h(W_n^T X). \quad (2)$$

When $W_i = W_j$, the activation z can be written as:

$$z^{'} = a_1 h(W_1^T X) + \cdots + (a_i + a_j) h(W_i^T X) + \cdots + a_n h(W_n^T X) \tag{3}$$

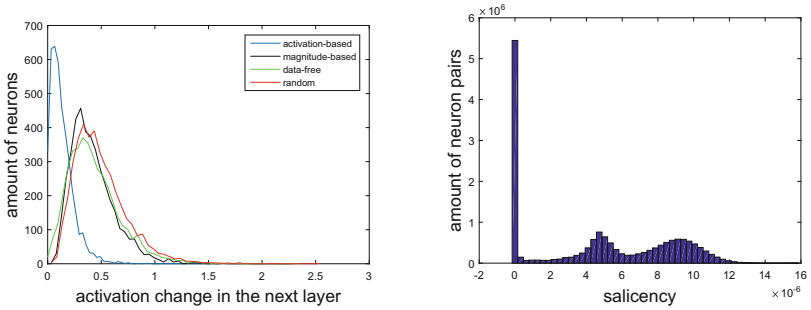
Equation(3) shows whenever two weight sets are equal, one of them can be effectively remove without causing any change in the activations of neurons in the next layer. $h(x)$ is the non-linear activation function. In this paper we use the ReLU function, so that $h(x) = max(0, x)$.

If we keep pruning weights without causing any change in the next layer, then the output of the network will remain the same, so that the recognition accuracy will be well retained as the size of model shrinks.

However, in large scale neural networks, a weight set in fully-connected layers contains a large amount of parameters, and there may not be two weight sets which are exactly the same. When $W_i \neq W_j$, the change of the activation is:

$$(z - z^{'})^2 = a_j^2 \left(h(W_j^T X) - h(W_i^T X) \right)^2 . \tag{4}$$

As the above equation shows, the change of the activation is no longer zero. In order to minimize it, we can find two most similar weight sets which minimize (4), and remove one of the neuron accordingly. Then we come to the definition of saliency as (1). To see how this method works, we evaluate the saliencies between every possible pair of neurons in fc6 layer, and we get the distribution of saliencies as Fig. 2(b) shows. Except for a peak around zero, there are many neuron pairs with large saliency values. When pruning process proceeds to remove those neurons, considerable change in the activations of the next layer becomes inevitable. This observation reveals that there is still much room for better maintaining the activation.



(a) Activation change after pruning (b) Saliency of all possible neuron pairs

Fig. 2. Activation change after pruning

3.2 Activation-Based Criterion for Further Maintaining the Activations

In this part, instead of pruning the whole neuron, we propose an activation-based criterion to prune weights within a neuron, keeping a small amount of weights to compensate for the change in the activation. By keeping a small amount of parameters, for example 1%, we can remove 99% parameters in W_j and obtain the sparsified weight set $W_j^{'}$. The change in activation becomes:

$$a_j^2 \left(h(W_j^T X) - h(W_j^{'T} X) \right)^2 \tag{5}$$

When $h(W_j^T X) = h(W_j^{'T} X)$, the change of activation is minimized. It is known that $h(W_j^T X) \geq 0$. When $h(W_j^T X) = 0$, we can simply set $a_j = 0$. When $h(W_j^T X) > 0$, we have:

$$h(W_j^T X) = \sum_{i=1}^{n} w_i x_i. \tag{6}$$

In the above equation, $w_1, \ldots, w_n$ and $x_1, \ldots, x_n$ represent the elements in the weight sets and input vectors, respectively. We can use a relatively small dataset to evaluate

$$E\left(h(W_j^T X)\right) = \sum_{i=1}^{n} E(w_i x_i) \tag{7}$$

Parameter w_i of bigger $abs\left(E(w_i x_i)\right)$ contributes more to the activations of neurons in the next layer. Therefore, we can sort w_i by their $abs\left(E(w_i x_i)\right)$ value in the descending order, reserve the first 1%, and remove the rest of parameters in W_j. Then we get the sparsified weight vector $W_j^{''}$. We use the dataset to calculate $E\left(h(W_j^{''T} X)\right)$. Let

$$W_j^{'} = \frac{E\left(h(W_j^T X)\right)}{E\left(h(W_j^{''T} X)\right)} W_j^{''} \tag{8}$$

Finally we obtain

$$E\left(h(W_j^T X)\right) = E\left(h(W_j^{'T} X)\right) \tag{9}$$

This method can better maintain the activations of the next layer than simply removing the whole neuron at a time. To verify this point, we estimate and compare the activation changes in fc7 layer, after 99% of the parameters in the fc6 layer are pruned using different pruning methods. Figure 2(a) shows the distribution of the activation changes of all neurons in fc7 layer after pruning. The method that removes one neuron at a time is referred to as data-free on account of requiring no training data to perform pruning, and is described in detail in [12]. Our method is referred to as activation-based. The other two methods are naive pruning techniques by removing weights randomly or removing those weights with small magnitudes. We can see from Fig. 2(a) that our

method is significantly better than the other methods in maintaining the activations of the next layer. The pruning procedure that we propose is summarized in Algorithm 1. The order of the list of neurons to be pruned is the same as the data-free method.

Algorithm 1. The proposed algorithm for weight pruning

Input: A list of weight sets W_i that connect the neurons to be pruned to the previous layer.

Start from the first neuron in the input neuron list.

1: **while** the requested compression rate is not reached **do**
2: Sort all the weights in W_i by the weight significance criterion described in Sect. 3.2 in the descending order.
3: Keep the first 1% weights and drop the rest of the weights in W_i, and get the sparsified weight set $W_i^{''}$.
4: Evaluate $E\left(h(W_i^T X)\right)$ and $E\left(h(W_i^{''T} X)\right)$, and let

$$W_i^{'} = \frac{E\left(h(W_i^T X)\right)}{E\left(h(W_i^{''T} X)\right)} W_i^{''}$$

5: Move on to the next neuron to be pruned.
6: **end while**
7: Retrain or fine-tune the pruned model if requested.

Output:The sparsified weight matrices $W_i^{'}$.

As illustrated above, we propose an activation-based weight significance criterion as well as a pruning procedure (see Algorithm 1). We believe that this algorithm can also be combined with other existing criterions to pursue other expectations. Here we present this point by introducing the correlation criterion into this procedure.

In [13], the authors found that connections that are strongly correlated have stronger predictive power for the activities of the latter layers in a deep architecture. A parameter or weight in neural networks represents a connection of two neurons in two adjacent layers. We compute correlations in the connections and sort them in a descending order, and remove those connections with small correlation values. In fact, we just simply replace activation-based criterion with correlation-based criterion in step 2 of Algorithm 1. By doing so, we can see if this algorithm can work well with other criterions and also compare our activation-based criterion with the correlation-based criterion.

3.3 Testing and Fine-Tuning

We use a pre-trained model and the pruned models to deal with the face verification task on the LFW face dataset [18], and compare the results by following the standard evaluation protocol. 6000 face pairs are evaluated to decide whether

the two faces in a pair belong to the same person. The last fully-connected layer fc8 is removed during testing, and the output 4096-dimensional vectors of the fc7 layer are extracted as features of the corresponding faces. Every face image is first rescaled to three different sizes: 256×256, 384×384, and 512×512. For each size, five patches of 224×224 pixel are cropped from the four corners and the center of the image with horizontal flip. As in [2], these 30 cropped image patches are input to the network, and 30 feature vectors of the face are extracted and averaged. The final averaged feature vectors will be compared by using the Euclidean distance.

A pre-trained baseline model would be used in our pruning experiments. Certain amount of parameters of certain layers will be pruned, and then the pruned model would be fine-tuned when requested. A deep neural network would converge to a minimum point during training. Any vibration on the parameters of the trained network would deviate the network from the minimum point. The motivation of minimizing the change in activations of the next layer is that it helps to keep the trained model remaining close to the minimum point. However, the optimal status of the trained network would be sabotaged inevitably by pruning operations. Fine-tuning the pruned network tries to make it converge to a minimum point again.

The collection of the training dataset is described in detail in [17]. Although the training dataset contains a massive amount of face images, the image quality of this dataset is not of satisfaction. So we used an augmented subset of the training dataset to fine-tune the pruned model. This dataset contains face

Table 1. Verification accuracy on LFW after pruning (the data-free method performs pruning in both of the fc6 and fc7 layers since it removes the whole neurons)

fc6 pruned	Random	Magnitude-based	Activation-based	Data-free [12]
0.25	0.9606	0.9613	0.9613	0.9613
0.50	0.9583	0.9611	0.9573	0.9531
0.75	0.9555	0.9573	0.9572	0.9420
0.99	0.6716	0.9303	0.9483	0.9202

Table 2. Verification accuracy on LFW after pruning and fine-tuning (pruning operation is performed on the fc6 layer except for the data-free method, which removes the whole neurons)

Weights pruned	Random	Magnitude-based	Activation-based	Correlation-based	Data-free [12]
97 MB	0.9743	0.9694	0.9737	0.9723	0.9730
194 MB	0.9707	0.9743	0.9720	0.9758	0.9712
291 MB	0.9680	0.9716	0.9703	0.9700	0.9697
384 MB	0.9510	0.9649	0.9657	0.9547	0.9644

images of 300 identities. These images were detected for face regions using the Viola&Jones face detector [20]. The faces were cropped out along with part of the background and then resized to the same scale. Erroneous faces were further filtered manually. We used this augmented dataset to retrain the pruned model for 5 epochs, and the learning rate is set to decrease from 10^{-4} to 10^{-6}.

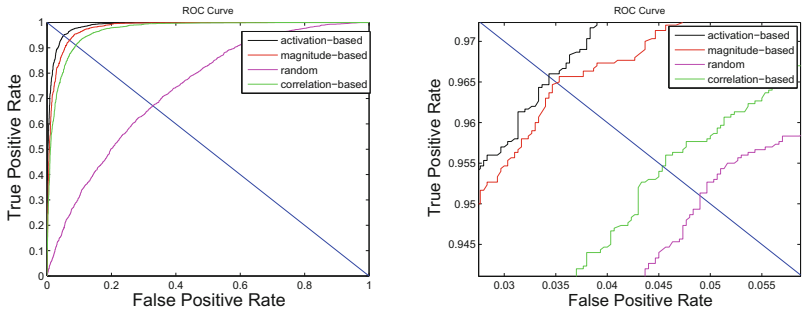
4 Experimental Results

The pre-trained baseline model that we use in our pruning experiments is trained and released by the group of [17]. When it is tested on the LFW verification task, it reports an accuracy of 96.13%. Experimental results of various pruning methods performed on the pre-trained model are shown in Table 1. The pruning procedures are carried out on the fully-connected layer fc6. As it is shown in Table 1, when 25% of the parameters in the fc6 layer are pruned using the magnitude-based or the activation-based method, the verification accuracies of the network do not degrade at all. When 50% or 75% of the parameters are pruned, the activation-based method does not show its edge over the other two methods. However, after pruning 99% parameters which means compressing the fc6 layer by 100 times, the model pruned with the activation-based method achieves a significantly better accuracy than those of the other methods. We also conduct experiments using the method in [12], which prunes a whole neuron (two layers related) at a time. When it removes the same amount of parameters in the two layers as the other methods do in one layer, it achieves accuracies of 0.9613, 0.9530, 0.9420, and 0.9202, which cannot compare to the proposed activation-based method.

Table 3. Experimental results after pruning two fc layers and fine-tuning

fc6 pruned	fc7 pruned	Activation-based	Correlation-based	Data-free [12]
194 MB	32 MB	0.9605	0.9694	0.9611
291 MB	48 MB	0.9704	0.9610	0.9590
384 MB	32 MB	0.9549	0.9406	–
384 MB	63 MB	0.9547	0.9268	0.9430

Experimental results after fine-tuning the pruned models are shown in Table 2. As we can see, after fine-tuning, the results of all pruned models are improved. The best accuracy achieved by the correlation-based method after fine-tuning is up to 97.58%, while the best results of other methods are around 97.4%, which suggests that the correlation criterion provides a better initialization for fine-tuning when certain amount of weights are removed. Also, these results in a way resonate the idea that sparse networks lead to better generalization. However, when it comes to removing 99% of the parameters in fc6 (384 MB), the activation-based method still gets the best result, which means



(a) Before fine-tuning. (b) After fine-tuning.

Fig. 3. ROC curves of pruned models, where 99% of parameters of the fc6 layer are removed.

that this method is a better choice for performing deeper pruning. The ROC curves of the models pruned by different methods are shown in Fig. 3. From these ROC curves we can know the advantage of the activation-based method when pruning a large amount of parameters of the network. Removing 99% of parameters in fc6 using the proposed method, we can reduce the model size from 553 MB to 128 MB, and the pruned model achieves an accuracy of 96.57%, which is even higher than the baseline model.

We have also conducted experiments on pruning both of the fc6 layer and the fc7 layer at the same time. And the results are shown in Table 3. We can see that the activation-based method performs significantly better than the correlation-based method and the data-free method, when it comes to pruning two fc layers at the same time. Even when both of fc6 and fc7 layers are deeply pruned using the activation-based technique, which amounts to compressing the network size from 553 MB to 65 MB, there is still only a slight drop in accuracy.

Compared to the study in [17], which improves the verification accuracy to 97.27% by increasing the depth of the network architecture, in this paper, we improve the accuracy to around 97.4% by sparsifying the network structure, and reduce the model size at the same time.

Table 4. The computational speeds

Model size	Total time	Time per face	Total time (GPU-accelerated)	Time per face (GPU-accelerated)
553 MB	193.81 s	0.48453 s	18.65 s	0.04662 s
65 MB	185.78 s	0.46445 s	12.18 s	0.03045 s

After the baseline model is pruned, the size of the model shrinks greatly. In the meantime, the decrease in the amount of parameters reduces the total times of addition and multiplication needed in forward propagation, which can speed

up the recognition process theoretically. So we compare the speeds of the pruned model and the baseline model by using them to extract face features from 400 face images. The experimental results are displayed in Table 4, which shows that the processing time needed for the pruned model is less than the baseline model. Especially when GPU is used, the speed is improved by 34.68%. However, the improvement in speeding up the recognition process is not as distinct as reducing the model size, because although the fully-connected layers contain most of the parameters, the convolutional layers require most of the computations.

The above experimental results show that the proposed method can deeply compress the size of deep neural networks without causing much loss in the recognition accuracy. Also, it is beneficial for speeding up the recognition process. Therefore, it has the potential to make deep neural networks easier to implement in many real applications.

5 Conclusion

In this paper, we propose an activation-based criterion to evaluate the importance of weights in deep convolutional neural networks. The proposed criterion is applied to a deep CNN trained for face recognition. We have carried out a series of experiments to verify the effectiveness of the proposed method. It is demonstrated that when the pre-trained baseline model is deeply compressed, the proposed method achieves the best performance, indicating that the activation information can be a useful indicator for estimation of weight significance. The study reported in this paper provides insights for reducing deep CNNs for face recognition. We believe that the proposed method is applicable in many deep learning scenarios independently or when combined with other methods.

Acknowledgements. This work was supported by National Natural Science Foundation of China (Nos. 61172141, U1611461, 61573387), Special Program for Applied Research on Super Computation of the NSFC-Guangdong Joint Fund (the second phase), Project on the Integration of Industry, Education and Research of Guangdong Province (No. 2013B090500013), Major Projects for the Innovation of Industry and Research of Guangzhou (No. 2014Y2-00213), and Science and Technology Program of Guangzhou (No. 2014J4100092).

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: International Conference on Neural Information Processing Systems, pp. 1097–1105 (2012)
2. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations, pp. 1–14 (2015)
3. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)

4. Szegedy, C., Liu, W., Jia, Y., et al.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
6. LeCun, Y., Denker, J.S., Solla, S.A.: Optimal brain damage. In: Advances in Neural Information Processing Systems, pp. 598–605 (1989)
7. Liu, C., Zhang, Z., Wang, D.: Pruning deep neural networks by optimal brain damage. In: Interspeech, pp. 1092–1095 (2014)
8. Anwar, S., Hwang, K., Sung, W.: Structured pruning of deep convolutional neural networks. ACM J. Emerg. Technol. Comput. Syst. **13**(3), 32.1–32.18 (2017)
9. Wolfe, N., Sharma, A., Drude, L., Raj, B.: The incredible shrinking neural network: New perspectives on learning representations through the lens of pruning (2017). arXiv preprint arXiv:1701.04465
10. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding. In: International Conference on Learning Representations, pp. 1–14 (2016)
11. Hassibi, B., Stork, D.G.: Second order derivatives for network pruning: optimal brain surgeon. In: Advances in Neural Information Processing Systems, pp. 164–171 (1992)
12. Srinivas, S., Babu, R.V.: Data-free parameter pruning for deep neural networks. In: British Machine Vision Conference, pp. 31.1–31.12 (2015)
13. Sun, Y., Wang, X., Tang, X.: Sparsifying neural network connections for face recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 4856–4864 (2016)
14. Denil, M., Shakibi, B., Dinh, L., et al.: Predicting parameters in deep learning. In: Advances in Neural Information Processing Systems, pp. 2148–2156 (2013)
15. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. In: British Machine Vision Conference, pp. 1–12 (2014)
16. Gong, Y., Liu, L., Yang, M., Bourdev, L.D.: Compressing deep convolutional networks using vector quantization (2014). arXiv preprint arXiv:1412.6115
17. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition. In: British Machine Vision Conference, pp. 41.1–41.12 (2015)
18. Huang, G.B., Mattar, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: a database for studying face recognition in unconstrained environments, Technical report 07–49. University of Massachusetts, Amherst (2007)
19. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: a unified embedding for face recognition and clustering. In: The IEEE Conference on Computer Vision and Pattern Recognition, pp. 815–823 (2015)
20. Viola, P., Jones, M.: Robust real-time face detection. Int. J. Comput. Vision **57**(2), 137–154 (2004)